DTL

The Data Transformation Language – which converts messages from one form to another.

Ensemble is based on message flow, and a data transformation is a way to convert from one message type to another. DTL (Data Transformation Language) adds a layer to this – it provides a graphical way to do the conversion. This is really helpful because most of the time, people with domain-specific knowledge may not have extensive coding skills. However, you always have the ability to do some coding, so if you need or want to, this is available.

DTL has several components: the data transformation engine, the language itself, and the DTL editor.

General Questions

Q: Is DTL a product?

A: No, DTL is not a product. It's an integral part of Ensemble.

Q: Where do I start with DTL?

A: To create and modify data transformations, from the Management Portal, go to Ensemble >> Build >> Data Transformations.

Q: What do I need to know before using DTL – are there prerequisites such as XML or Caché ObjectScript?

A: All communications within Ensemble goes on via messaging. Once you have your messages, there are no technical prerequisites to using the DTL editor. You only need to understand the content of the messages that you're transforming, so that you can be clear about what's in the messages and what you're going to do with them.

Q: Where can I use DTL?

A: DTL is most often used from a business process written in BPL or a routing rule. A data transformation can be used from any server-side code, such as Caché ObjectScript, because it is just a class which implements a Transform method.

Q: Can I edit DTL from a browser?

A: Yes! Starting with 2011.2, you can edit BPL and DTL from the Management Portal as well as with Studio.

Q: Can I put DTL into source code control?

A: Yes, Studio and the Management Portal use a common source control framework. For more information, see the "Using Studio Source Control Hooks" chapter in *Using Caché Studio*.

Q: Where do I learn more?

A: To read more about DTL, start with the "Production Data" chapter of *Developing Ensemble Productions*. For more details about the language elements, see the *Ensemble Data Transformation Language Reference*.

Development Questions

Q: Why use DTL instead of code?

A: Because it's easy! It's accessible – you don't have to be a programmer to understand what it's doing. You get a visual representation of what's happening in your transformation, instead of having to read through code. This is especially useful in situations where an analyst has domain-specific knowledge, rather than programming skills.

Q: How do I access details of the business process from inside DTL, and what variables are available in DTL?

A: There are source and target variables available that you can use directly (not surprisingly, these reference the source and target messages).

You can also create local variables to use in your DTL. You can do this either in a <code> block or an <assign> statement.

If a DTL is called from a business process (such as a message router), you have access to the process variable, which is a reference to the business process, and the context variable, which is a reference to business process context. The context variable is an object instance with properties specified by the application developer.

For more information on this, see the section "Business Process Execution Context" in the "Business Processes" chapter of *Developing Ensemble Productions*.

Q: What data structures can I manipulate with DTL?

A: You can manipulate any class or virtual document, including HL7 messages, XML virtual documents, or even classes such as Sample.Person. The source and the target of the DTL can be completely different.

Q: When would I use "create = existing"?

A: DTL allows you to:

- Create a copy of the original and edit it (create = copy)
- Create an empty instance of the target (create = new)
- Modify an existing message in place (create = existing)

The most common reason for using create=existing is to apply multiple DTLs to the same message. The first DTL would create a new message and then subsequent transformations would be made on that message. For example, a routing rule can specify multiple DTLs for same destination, separated by commas.

Q: Can I use class inheritance to design data transformations?

A: No. You cannot add extra actions to a DTL by making a subclass, because actions in the subclass completely replace the actions in the superclass.

Q: What is a subtransform and when would I use one?

A: A subtransform is a normal DTL being called from another DTL. It provides a way to reuse common transform code. For example, you may need to manipulate your PID segments uniformly for many HL7 transformations. In this case, you can create a DTL to do this, which is called as a subtransform.

Q: How do I put comments in my DTL?

A: In the DTL editor, on the Action tab, use the Description field. For those familiar with the XML, this is stored in the <annotation> tag.

Q: How do I loop over repeating segments, collections, etc.?

A: A simple assignment with empty parentheses will implicitly copy the entire collection. If you want to manipulate each item in the collection, then use a For-Each action to iterate over the group. A For-Each action will work on a property that is followed by (), which indicates that the property is a collection or repeats.

For other properties that contain multiple values, such as $List or a delimited string, use a code block.

Q: How do I conditionally change the value of my message in DTL?

A: You use the pull down to use an If action, the condition can be any Boolean expression, such as checking the value of an HL7 field.

Q: How do I do a lookup inside DTL?

A: For simple lookups, you can use the Lookup function and a lookup table, which is described in the "Lookup Tables" section of the "Creating a New Production" chapter of *Developing Ensemble Productions*.

You can create SQL lookups using the <sql> action to query data local to Ensemble. If you need to access data outside Ensemble, the correct approach is to create a BPL that fetches the information and stores it in the context before invoking the DTL.

(While it is possible to use the <sql> action with tables linked using the SQL Gateway, this is not recommended – it makes recovery from network failures very difficult.)

Q: What functions are available from DTL, and can I create my own?

A: A long list of available functions and operators is available in the DTL editor, such as ToLower and ToUpper. You can also call any class method with the regular ##class method call.

If you need to perform a task repeatedly and want to have the action available in the DTL editor, you can create your own function by creating a class method in a class that extends Ens.Rule.FunctionSet. For more information, see the "User-defined Functions" section of the "Creating a New Production" chapter of *Developing Ensemble Productions*.

Q: How do I handle very large segments (that is, over 32KB)?

A: HL7 messages with segments longer than 32KB need to be handled carefully.

If you are just copying the segment unchanged to the target, then nothing special is needed. But if you are modifying the segment, you need to use the ReadRawDataStream and StoreRawDataStream methods of the EnsLib.HL7.Segment class.

The long field in a very large segment is normally the last field in the segment. Fields that finish before the 32KB point can be accessed and written to the target segment in the normal way. But you must read the long field into a stream object using ReadRawDataStream and then write it to the target using StoreRawDataStream. If you want to modify the long field then you must manipulate the stream in a Code action.

In the rare case that there are fields after the long field, they will be read and stored along with the long field and you will have to handle them. StoreRawDataStream must be the last change you make to the target segment.

Q: How do I test my DTL?

A: In the DTL editor, use the Test button on the Tools tab. You can input content you're your source message.

If your source message is HL7 or another EDI (Electronic Data Interchange) format, you paste it into the input message box; for non-EDI messages, your input needs to be in XML format. For more information, see "Testing the DTL Data Transformation" section of the "Production Data" chapter of *Developing Ensemble Productions*.

Troubleshooting Questions

Q: Why does my HL7 transform work when I'm testing but not in production?

A: Check that you're not modifying the source message! By default, inbound HL7 messages in productions can't be modified.

If you're not modifying the source message, then make sure your HL7 schemas (such as 2.4) match in your production and your transformation.

Q: What are build map errors when I test my DTL?

A: A build map error shows that an HL7 message does not comply with the schema. For example, a required segment is missing or an unexpected segment is present. In your production configuration, you choose how to handle them. Ensemble always shows these errors when you view the message content.

Q: Why do I get "No segment found" errors?

A: When transforming HL7 messages and other documents, if you copy a value from the source segment and the source segment is not present, you will get an error. Prior to 2011.2, the default behavior was to not report errors in DTL; in 2011.2 and subsequent versions, errors are reported by default – and the DTL will fail.

To change the default behavior, clear the Report Errors check box on the Transform tab in the DTL editor. This corresponds to the REPORTERRORS parameter in the class definition.

Q: When I use the GUI to drag and drop repeating segments, why do I only get the first one?

A: The DTL editor puts a 1 inside any parentheses that appear in the property and value expressions of the <assign> action. This means that only the first occurrence will be copied. If you want all the values of a collection, then remove the 1 – from the property and the value.

For example, change

```
target.{ORCgrp(1)}
```

to

```
target.{ORCgrp()}
```

Q: Why do I get <STORE> errors with large messages?

A: As you loop over segments in an HL7 message or object collections, they are brought into memory. If these objects consume all the memory assigned to the current process, you may get unexpected errors.

To avoid this, remove the objects from memory after you no longer need them. For example, if you are processing many HL7 segments in a For-Each loop, you can call the commitSegmentByPath method on both the source and target as the last step in the loop. Similarly, for object collections, use the %UnSwizzleAt method.

If you cannot make code changes, a temporary workaround is to increase the amount of memory allocated for each process. You can change this by setting the bbsiz parameter on the Advanced Memory Settings page in the Management Portal. Note that this requires a system restart and should only occur after consulting with your system administrator.

Q: Where did my XML go?

A: Beginning with version 2011.2, the XML pane of the DTL editor has been replaced with an Action list. This improves usability in many ways. If you really want to, you can access the XML directly using Studio. Open the DTL and click the View Other Code button. This option is not available in the Management Portal.